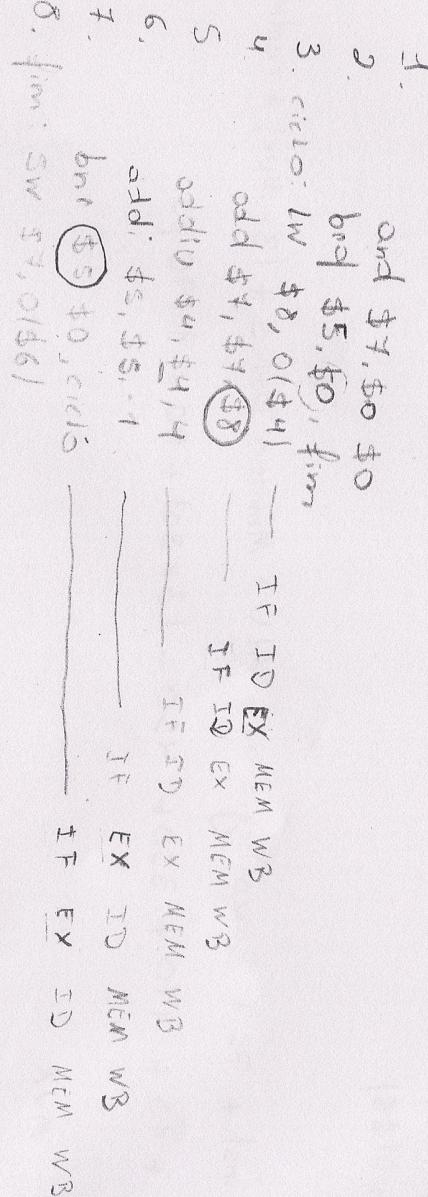


como se viene o venga el resultado que estan buscando en su trabajo. Es, por lo tanto, necesario unificar



Existem 2 dependências funcionais.

funcão add pressa do sensor
é feita ainda não estou desempenhando, e na
fase 2, comparamos o valor do sensor
com o valor que é o resultado da operação de
acumulação de resultados, e na
fase 3 quando o resultado da operação de
acumulação é igual ao resultado da operação
de comparação o valor do sensor é
atualizado para que seja o resultado da
operação de acumulação.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
odd	IF	LD	EX	MEM	WB									
bcd		JF	ID	EX	MEM	WB								
LW			JF	ID	EX	MEM	WB							
add				IF	***	ID	EX	MEM	WB					
add1					***	IF	ID	EX	MEM	WB				
odd:						IF	ID	EX	MEM	WB				
bnc							***	***	***	IF	ID	EX	MEM	WB

Nº andares = 7 → 500 necesarios 14 ciclos de
Nº ciclos = 14 ruedas.

500 rec

Arquitectura de Sistemas e Computadores II

1^a Frequência

Departamento de Informática ◊ Universidade de Évora

29 de Outubro de 2014

Indique todos os cálculos efectuados

Perguntas rápidas

1. [0,5 valores] Sabendo que a frequência do relógio do processador X, pode concluir que o desempenho de X é superior ao de Y?
2. [0,5 valores] Seja m uma instrução com um *delay slot*. A instrução no *delay slot* de m é executada sempre que m é executada?

3. [0,5 valores] Se a execução de um programa é interrompida devido à ocorrência de uma exceção ou de uma interrupção, o que acontece ao programa quando o tratamento da exceção ou da interrupção termina?

4. [0,5 valores] Que tipo de conflito existiria se um *issue packet* do pipeline MIPS com *double issue* estático contivesse duas instruções de acesso à memória?

Desempenho

5. [3 valores] Um programa é executado no processador A, cujo relógio tem um período de 0,5 ns. Na execução do programa são executados 1000 milhões de instruções, divididas por três classes de acordo com a distribuição seguinte:

Classe	Aritméticas	Saltos condicionais	Acesso à memória
CPI	1	2	3
%	60	20	20

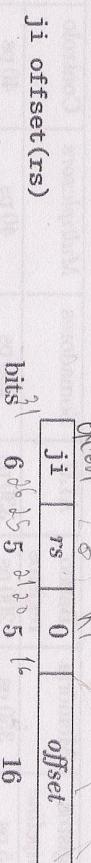
No processador B, que implementa a mesma arquitectura, foi possível baixar o período do relógio para 250 ps. No entanto, na execução do programa neste processador, verifica-se que o *tempo* que demoram as instruções de acesso à memória é *exactamente o mesmo* que demoram no processador A.

Qual o processador mais rápido para o programa e qual o *speedup* que ele oferece em relação ao outro?

Implementação MIPS monociclo

Para este grupo, use como referência a implementação monociclo da Figura 1.

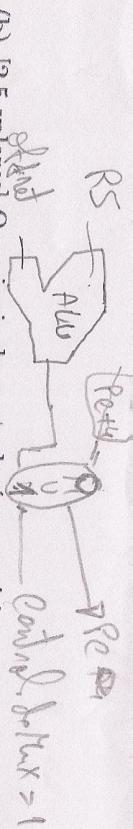
6. Pretende-se incluir a instrução *ji* (*jump indirect*) na implementação MIPS monociclo. A instrução *ji* é uma instrução tipo-I com dois argumentos:



O efeito desta instrução é provocar o salto da execução para a instrução cujo endereço é obtido somando offset, que pode ser negativo, ao conteúdo de rs.

- (a) [2,5 valores] Quais das unidades funcionais existentes serão usadas para a execução desta instrução e que unidades funcionais é necessário acrescentar?

*Mais uma unidade multiplicadora é necessária para o *mult*, já que o resultado é dividido entre 256. A unidade multiplicadora é necessária para dividir o resultado entre 256.*



- (b) [2,5 valores] Que sinais de controlo é necessário acrescentar e quais os valores que os vários sinais de controlo deverão ter para a execução desta instrução? (Não precisa de indicar o valor de ALUOp, basta dizer qual será a função da ALU na execução desta instrução.)

Se considerar necessário fazer alguma alteração ao caminho de dados, apresenta-a na Figura 1.

Pipeline MIPS de 5 andares

Para este grupo, use como referência o *pipeline* da Figura 2. Tenha, no entanto, em atenção as caracterizações do funcionamento do *pipeline* feitas nas várias alíneas.

7. [1 valor] Considere, agora, a inclusão da instrução *ji* na implementação *pipelined* do MIPS.

Em que andar estará a instrução no ciclo em que o endereço do destino do salto fica disponível? Quantos *delay slots* deveria ter esta instrução para que não fosse necessário introduzir atrasos nem apagar instruções do *pipeline*?

8. Pretende-se executar o código MIPS seguinte de modo a que o seu efeito seja exactamente o que teria se fosse executado na implementação monociclo do processador. No fim da execução do código, os valores nos registos usados não são importantes, excepto o do registo \$v0.

5: 5	\$v0	1. or \$v0, \$0, \$0	
3: 2	\$t0	2. ciclo: lw \$t0, 0(\$s1)	
4: 2	\$t0	3. sw \$t0, 0(\$s0)	
6: 6	\$t0	4. beq \$t0, \$0, fim	
7: 7	\$t0	5. addi \$v0, \$v0, 1	
2: 7	\$t0	6. addiu \$a1, \$a1, 4	
6: 3	\$t0	7. addiu \$a1, \$a1, 4	
5: 1	\$v1	8. beq \$0, \$0, ciclo	
		9. fim: jr \$ra	

- (a) [2 valores] Identifique todas as dependências (de dados) existentes no código apresentado.

(b) [2 valores] Simule a execução do código desde o início e até terminar a primeira execução da instrução 8. (Não precisa de considerar as instruções que entram no *pipeline* após o início da execução desta instrução.) Considere um processador com *forwarding*, com decisão dos saltos condicionais no andar EX, com previsão perfeita do resultado das instruções de salto condicional e sem *delay slots*. Apresente a evolução do estado do estudo do *pipeline* durante a execução, indicando todos os atrasos introduzidos e todos os pontos onde foi necessário o *forwarding* de algum valor, identificando claramente entre que andares o *forwarding* foi feito.

Quantos ciclos de relógio são necessários para executar o código nas condições acima?

Quantos ciclos de relógio seriam necessários para executar o código se fossem executadas 1000 iterações do ciclo?

- (c) [2 valores] Altere o código apresentado, reordenando as instruções e, se considerar útil, modificando o offset das instruções de acesso à memória, de modo a eliminar o maior número possível de atrasos e de ciclos desperdiçados durante a sua execução no *pipeline* com *forwarding*, com decisão dos saltos condicionais no andar ID e com um *branch delay slot*.

9. [3 valores] As latências das várias componentes do *pipeline* são as apresentadas na tabela seguinte:

PC	Registros do pipeline	Memória	Banco registos	ALU	Somadores	Multiplexors	Controlo	Controlo da ALU
10 ps	15 ps	350 ps	200 ps	330 ps	250 ps	40 ps	40 ps	20 ps

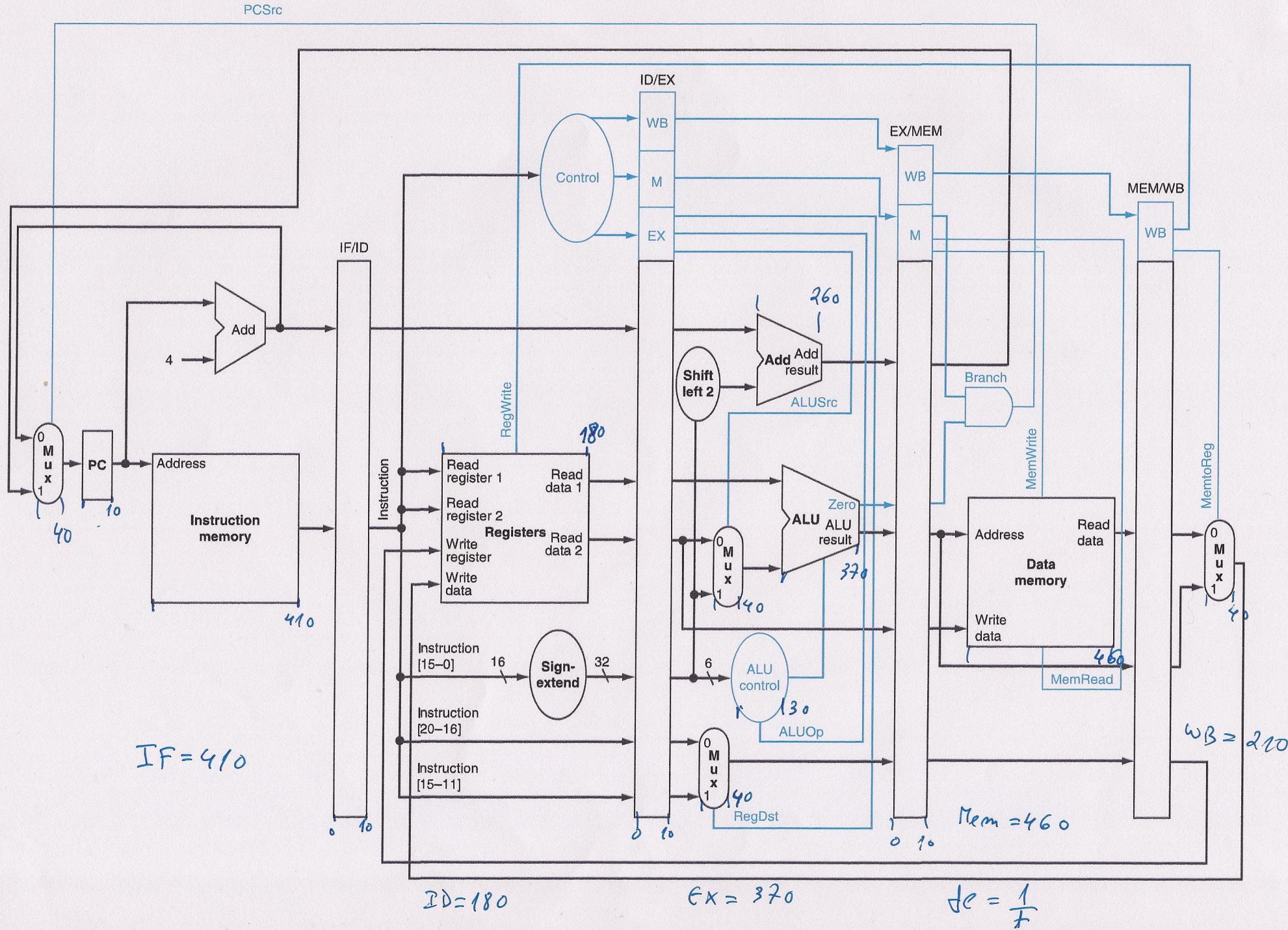
Considere que os restantes elementos lógicos têm latência zero.

Quais as unidades funcionais, de controlo, etc., que se encontram no caminho crítico de cada andar do *pipeline*? Qual a latência desses caminhos?

Qual o andar que determina o período mínimo do ciclo de relógio? Qual é esse período?

✓

Tempo



(5)

$$P_A = 1 \times 10^{-9}$$

$$P_B$$

$$Fe = 250 P_A = 250 \times 10^{-12} n$$

$$CPI_B = ?$$

$$Fe = 5 \times 10^{-10} n$$

$$(P_A) \cdot CPI_A = (0,60 \times 1) + (0,20 \times 2) + (0,20 \times 3) = \\ = 0,6 + 0,40 + 0,60 = 1,6$$

$$CPI_A = 1,60 \times 10^{-9} \times 1,6 = 1,0 \times 10^{-9}$$

$$J = \frac{1}{4}$$

$$J = 2 \times 10^{-9}$$

$$CPI_{Time A} = \frac{1 \times 10^{-9} \times 1,6}{2 \times 10^{-9}} = 0,8 \times 10^{-10} s$$

(P_B)

$$CPI_B = ?$$

$$n_{int.} = 1 \times 10^{-9}$$

$$Fe_B = 250 \times 10^{-10} n \Rightarrow J = \frac{1}{4} = 4 \times 10^{-9} / Hz$$

$$CPI_B = (0,60 \times 1) + (0,20 \times 2) + (0,20 \times 6) = 2,2$$

$$\overline{Time Exec A} = \frac{n_{int} \times CPI_A}{Fe_A} = \frac{4 \times 10^{-9} \times 0,8}{2 \times 10^{-9}} = \frac{3,2 \times 10^{-9}}{Fe_B}$$

Time Exec B

$$J = \frac{0,8}{\frac{6,4 \times 10^{-9}}{2 \times 10^{-9} CPI_B}} = \frac{CPI_B}{1,6 \times 10^{-9}} \Leftrightarrow \frac{CPI_B}{Time_B} = 4$$

$$J = \frac{0,8}{\frac{6,4 \times 10^{-9}}{4}} \Rightarrow T. CPI_B = 0,5 \times 10^{-9}$$

$$Modul = \frac{0,8}{0,55} = 1,45 \quad \text{Tanto de CPIB, mella præm 145 neg dr}$$

$$\text{dilution } \frac{1}{T} = \frac{1}{ID} + \frac{1}{ex} + \frac{1}{rec} + \frac{1}{BS} \quad (extinct + RS) = Rec + r$$

alinea g) dependencias

• i? i?

462
- instrucciones

19

$$N_{\text{cells}} = 116 \times 10^9$$

۷۱

$$N^{\text{rich}} = 1 \times 10^9 \times 0.6 + A + B$$

Arquitectura de Sistemas e Computadores II

1ª Frequência

Departamento de Informática
Universidade de Évora

28 de Outubro de 2015

Indique todos os cálculos efectuados

Perguntas rápidas

1. [0,5 valores] Se os processadores X e Y implementarem a mesma arquitectura, é possível comparar o seu desempenho para um programa conhecendo só os CPI para o programa e os períodos dos relógios?

2. [0,5 valores] O que permite que a leitura do conteúdo dos registos usados por uma instrução MIPS seja feita em simultâneo com a identificação da instrução? *lect de other fields or memory pages*

3. [0,5 valores] Se a instrução que está no andar MEM do pipeline MIPS gera uma exceção, o que acontece às 3 instruções que entraram depois dela no pipeline? *breakdown no next*

4. [0,5 valores] O pipeline de um processador tem 6 andares, cujos caminhos críticos têm latências 400 ps, 340 ps, 190 ps, 270 ps, 350 ps e 245 ps, respectivamente. Qual seria o andar que tentaria dividir, se quisesse construir um processador que pudesse funcionar com uma frequência de relógio superior à deste? *divisão a partir de 400ps*

Desempenho

5. [3 valores] O programa P é executado no computador A , cujo ciclo de relógio dura 1 ns, com um CPI de 2,2. Na execução do programa são executadas 500 milhões de instruções, com a seguinte distribuição:

Classe	Aritméticas	Acesso à memória	Saltos
%	40	30	30

(a) [1,5 valores] Qual o tempo de CPU necessário para a execução de P em A ? *34 ns*

(b) [1,5 valores] Numa outra implementação da mesma arquitectura, o computador B , o CPI das instruções de acesso à memória, que era 4 em A , foi reduzido para 3, mantendo-se tudo o resto na mesma. Qual o speedup obtido quando P é executado em B ? *1,25*

Implementação MIPS monociclo

6. Pretende-se que a implementação MIPS monociclo da Figura 1 suporte a execução da instrução lui (*load upper immediate*), que é uma instrução tipo-I com dois argumentos:

lui rt, immediate

lui	0	rt	immediate
bits	6	5	5
			16

Esta instrução coloca no registo rt o valor $immediate \times 2^{16}$ (= $immediate$ ~~dezasseis 0's~~). *immediate*

(a) [2,5 valores] Quais das unidades funcionais existentes serão usadas para a execução desta instrução e que unidades funcionais é necessário acrescentar?

Apresente, na Figura 1, as alterações que considerar necessário fazer ao caminho de dados.

- (b) [2,5 valores] Que sinais de controlo é necessário acrescentar e quais os valores que os vários sinais de controlo deverão ter durante a execução desta instrução?
 (Não precisa de indicar o valor de ALUOp, basta dizer qual será a operação executada pela ALU durante a execução desta instrução.)

Pipeline MIPS de 5 andares

Para este grupo, use como referência o *pipeline* da Figura 2. Tenha, no entanto, em atenção as caracterizações do funcionamento do *pipeline* feitas nas várias alíneas.

7. [1,5 valores] Considere, agora, a inclusão da instrução *lui* na implementação *pipelined* do MIPS.

Visto o valor a escrever no registo *rt* poder ser calculado sem recurso à ALU, ele poderia ser lá escrito quando a instrução estivesse no andar ID.

Apresente uma razão para isso não acontecer. Em que andar deverá estar a instrução no ciclo em que o valor é escrito no registo?

8. Considere que o significado e o efeito do código MIPS seguinte são exactamente aqueles que teria se fosse executado na implementação monociclo do processador (onde não existem *delay slots*). No fim da execução do código, o único registo usado cujo valor é importante é o registo \$v0.

1. *lw* \$v0, 0(\$a0)
 2. *or* \$t0, \$0, \$0
 3. *lw* \$t1, 0(\$a1)
 - 4. *beq* \$t0, \$t1, *fim* *100 X 5*
 5. *ciclo:* *lw*, \$t2, 4(\$a0),
 6. *addiu* \$a0, \$a0, 4
 7. *sw* \$t2, -4(\$a0)
 8. *addi* \$t0, \$t0, 1
 - 9. *bne* \$t0, \$t1, *ciclo*
 10. *fim:* *addi* \$t1, \$t1, -1
 11. *sw* \$t1, 0(\$a1)
 12. *jr* \$ra

- (a) [2 valores] Identifique todas as dependências (de dados) existentes no código apresentado.

- (b) [3 valores] Simule a execução do código, quando os saltos condicionais das instruções 4 e 9 não são efectuados, num processador com *forwarding*, com decisão dos saltos condicionais no andar ID, com previsão perfeita do resultado das instruções de salto condicional e sem *delay slots*. Apresente a evolução do estado do *pipeline* durante a execução, indicando todos os atrasos introduzidos e todos os pontos onde foi necessário o *forwarding* de algum valor, identificando claramente entre que andares o *forwarding* foi feito.

Quantos ciclos de relógio são necessários para executar o código nas condições acima? *19*

Quantos ciclos de relógio seriam necessários para executar o código se o ciclo (instruções 5 a 9) fosse executado 100 vezes?

- (c) [2 valores] Altere o código apresentado, reordenando as instruções e, se considerar útil, modificando o *offset* das instruções de acesso à memória, de modo a eliminar o maior número possível de atrasos e de ciclos desperdiçados durante a sua execução no *pipeline* com *forwarding*, com decisão dos saltos condicionais no andar ID e com um *branch delay slot*.

ILP

9. [1,5 valores] Diga, justificando, se será possível um *issue packet* do *pipeline* MIPS com *double issue* estático ser constituído pelas duas instruções

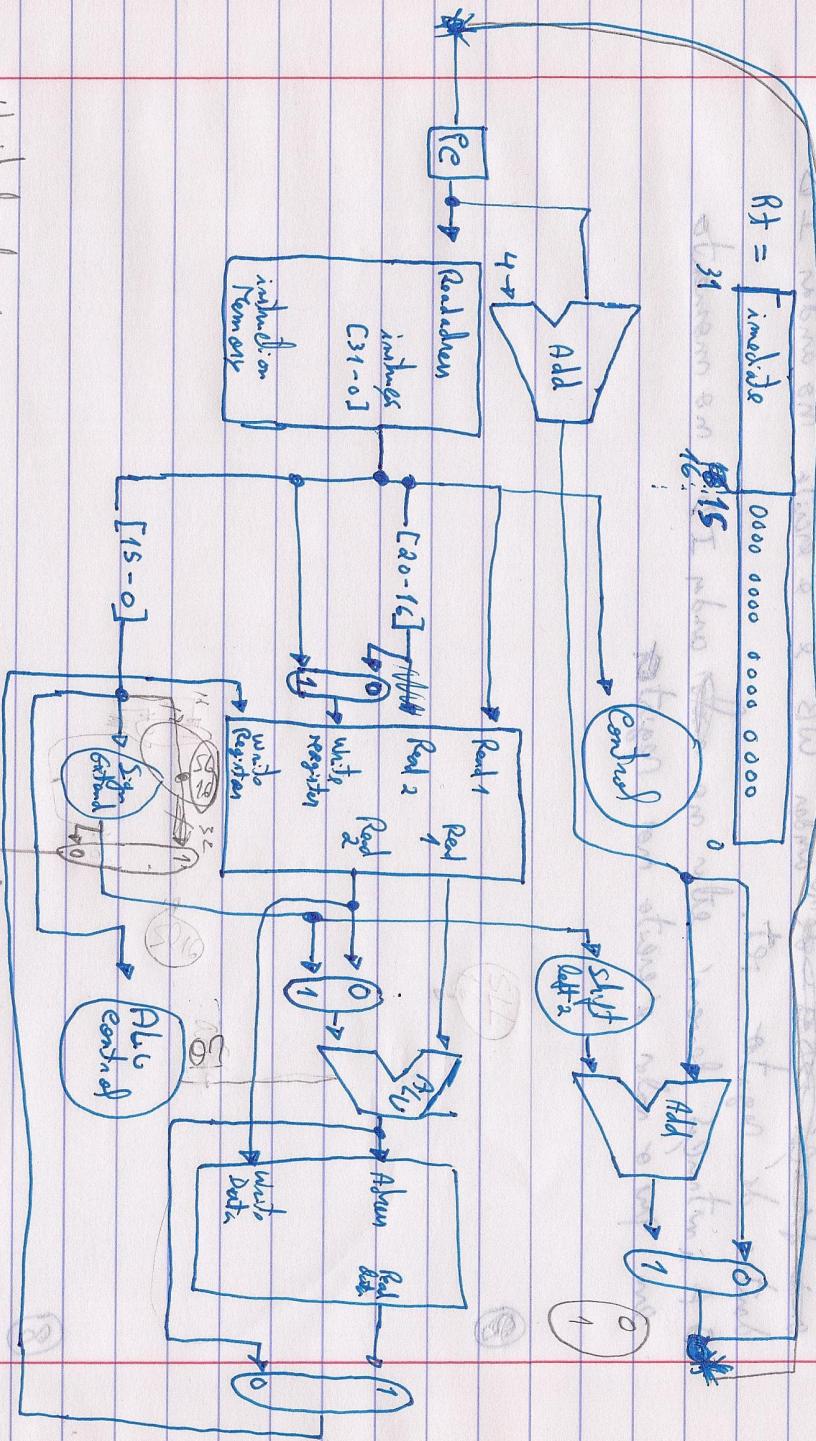
add \$1, \$2, \$3 e lw \$1, 0(\$4).

$$CPT_{global} = (0, 40) + (\beta \times 0, 30) + (\alpha, 30) = 1,6$$

⑥ Jui. Rt , immediate value o' modo load immediate

$Rt = imm < 16$

00111	0	met	Address 16 bits
31	2625	2120	16:15



Unidimensionais

Controle de mux

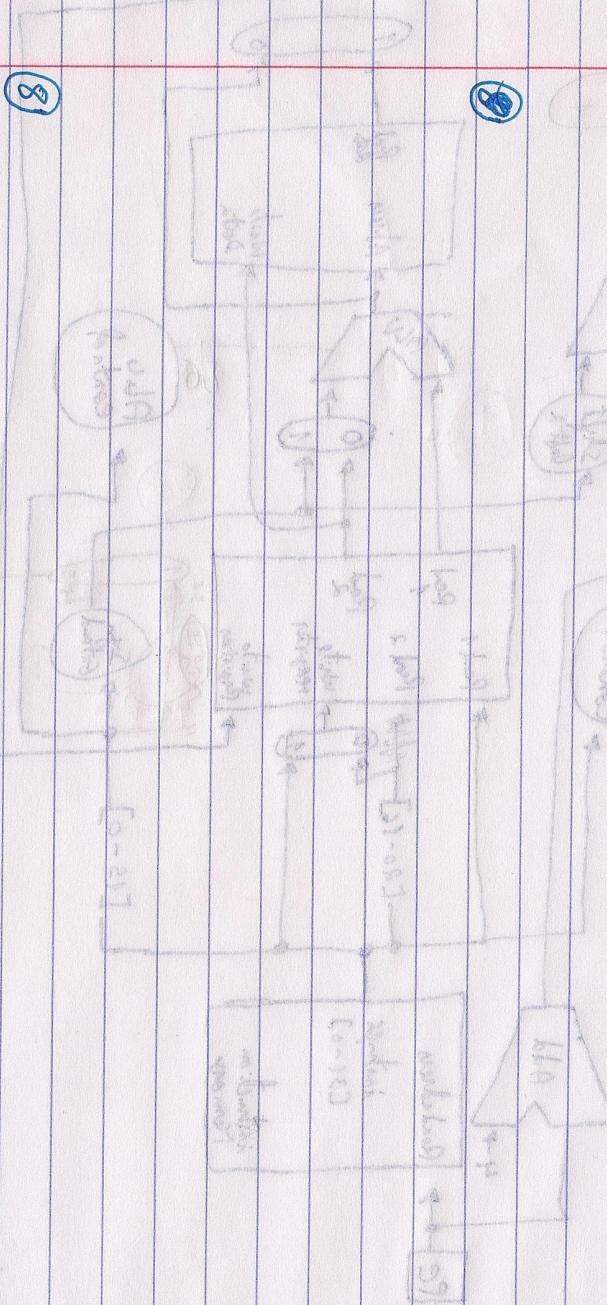
É necessário usar o mesmo SLL de 16 bits a zero considerada função de multiplicador para dividir MUX & saida de registo extendido

Controle do SLL 16 bits far. out. simétrico
Controlador de mux a "1" para este instruções

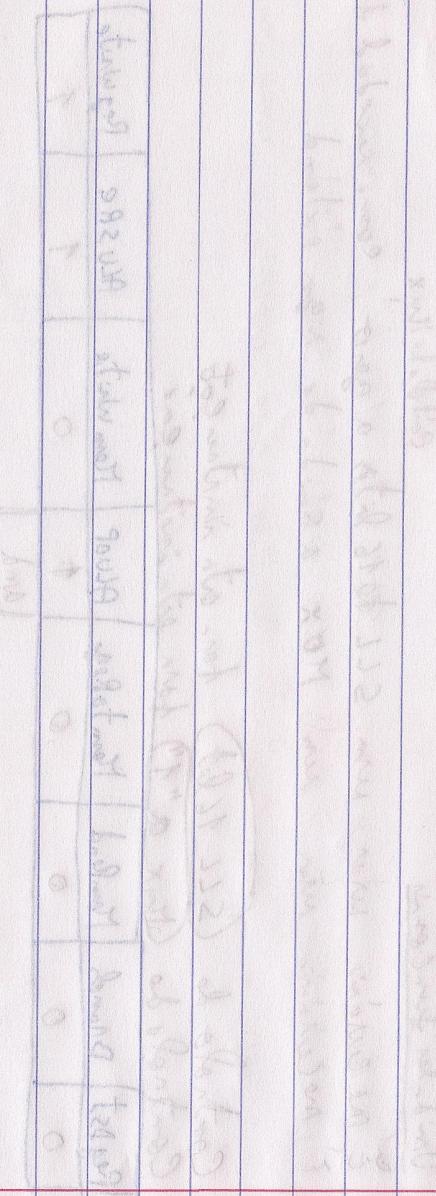
Regst	Branch	MemRead	MemToReg	ALUOp	MemWrite	ALUOp	Regwrite
0	0	0	0	1	0	1	1

and

- 7) ~~O registrador RT guarda o resultado da operação imediata + 16 bits que são extraídos de uma das entradas de saída de menor peso e inseridos no endereço WB e escrita no endereço ID de destino de saída de menor peso.~~
- 8) A instrução dezena é escrita em ID no momento em que o valor é escrito no registrador.



8)



⑧e

lw

07

lw

08

beq

09

not

0A

add

0B

add

0C

sw

0D

add

0E

lmg

0F

~~addi not~~

~~add~~

10

sw

11

nop

12

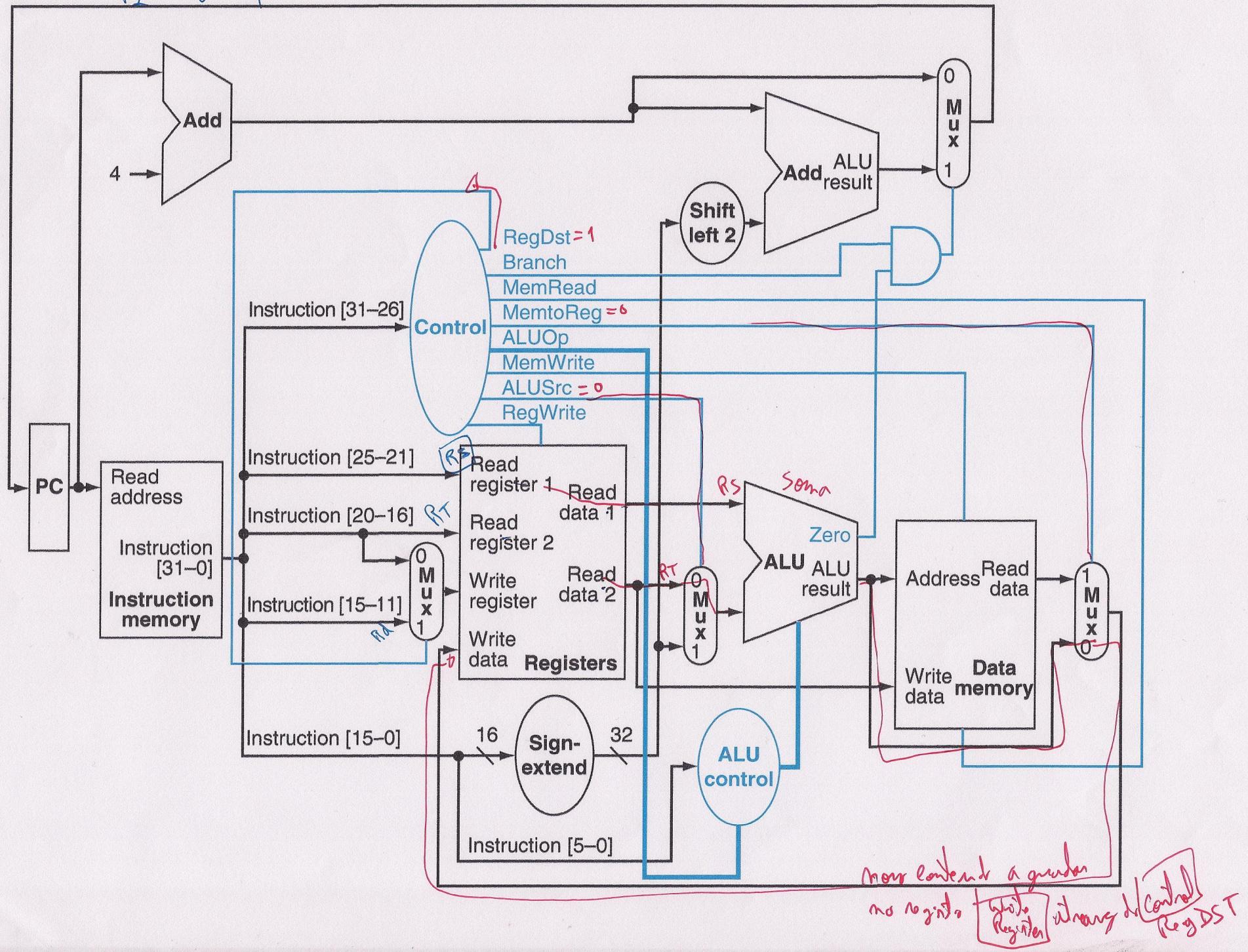
Im: R+, imm # R+ = imm << 16

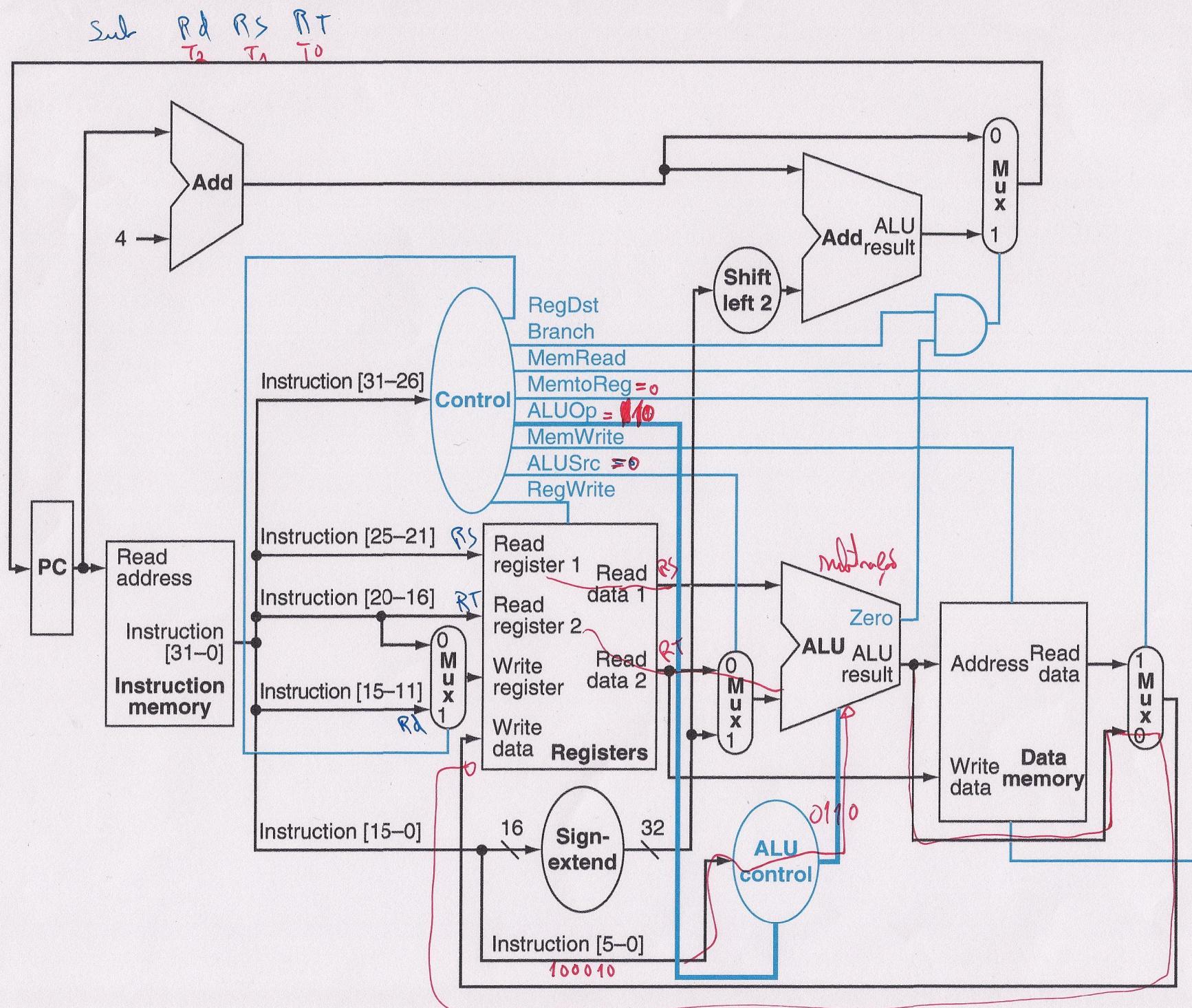
OpCode	RS	RT	imm
1001111	000000	R+	16 bits

for T

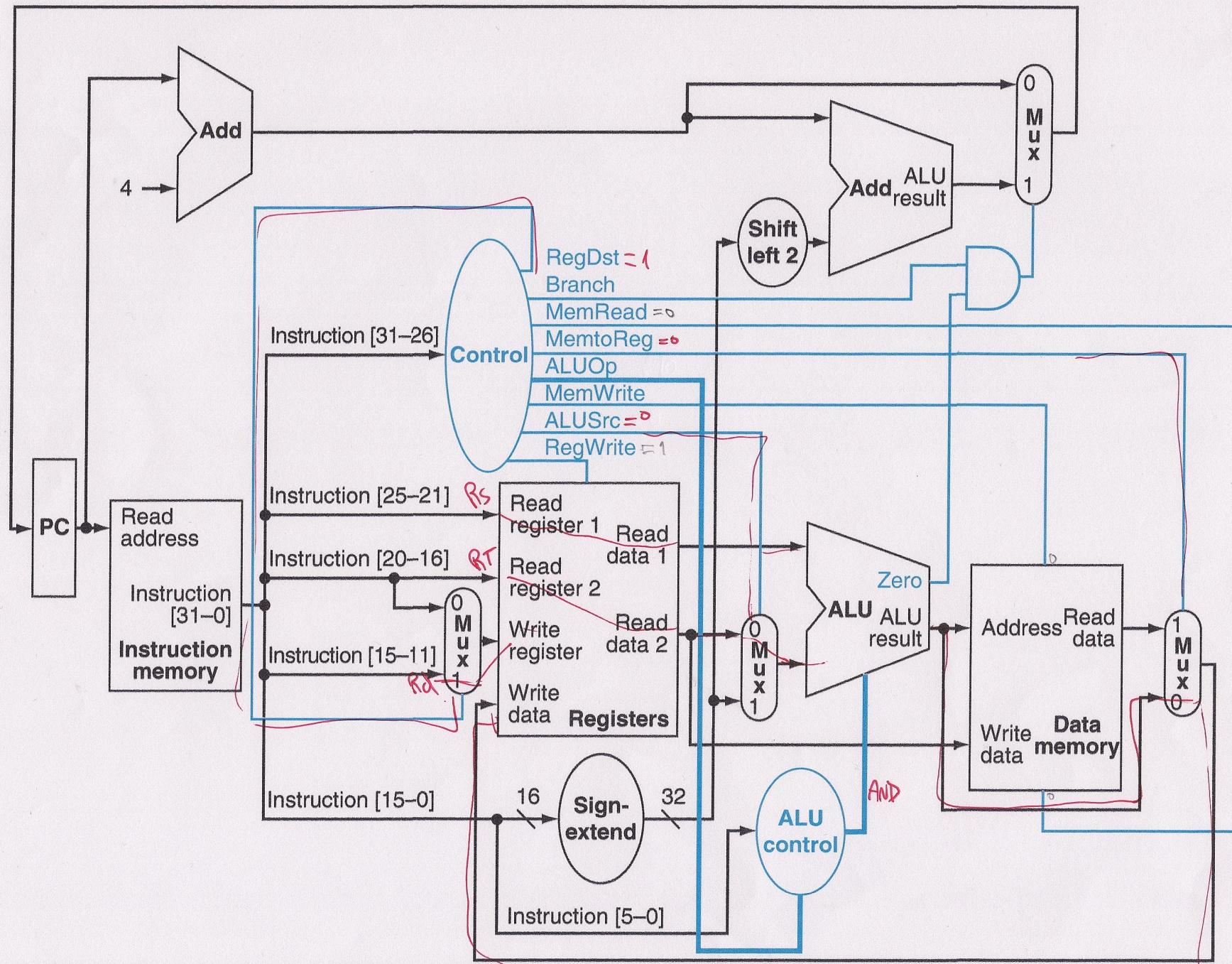
31

add RD, RS, RT
T₂ T₀ T₁

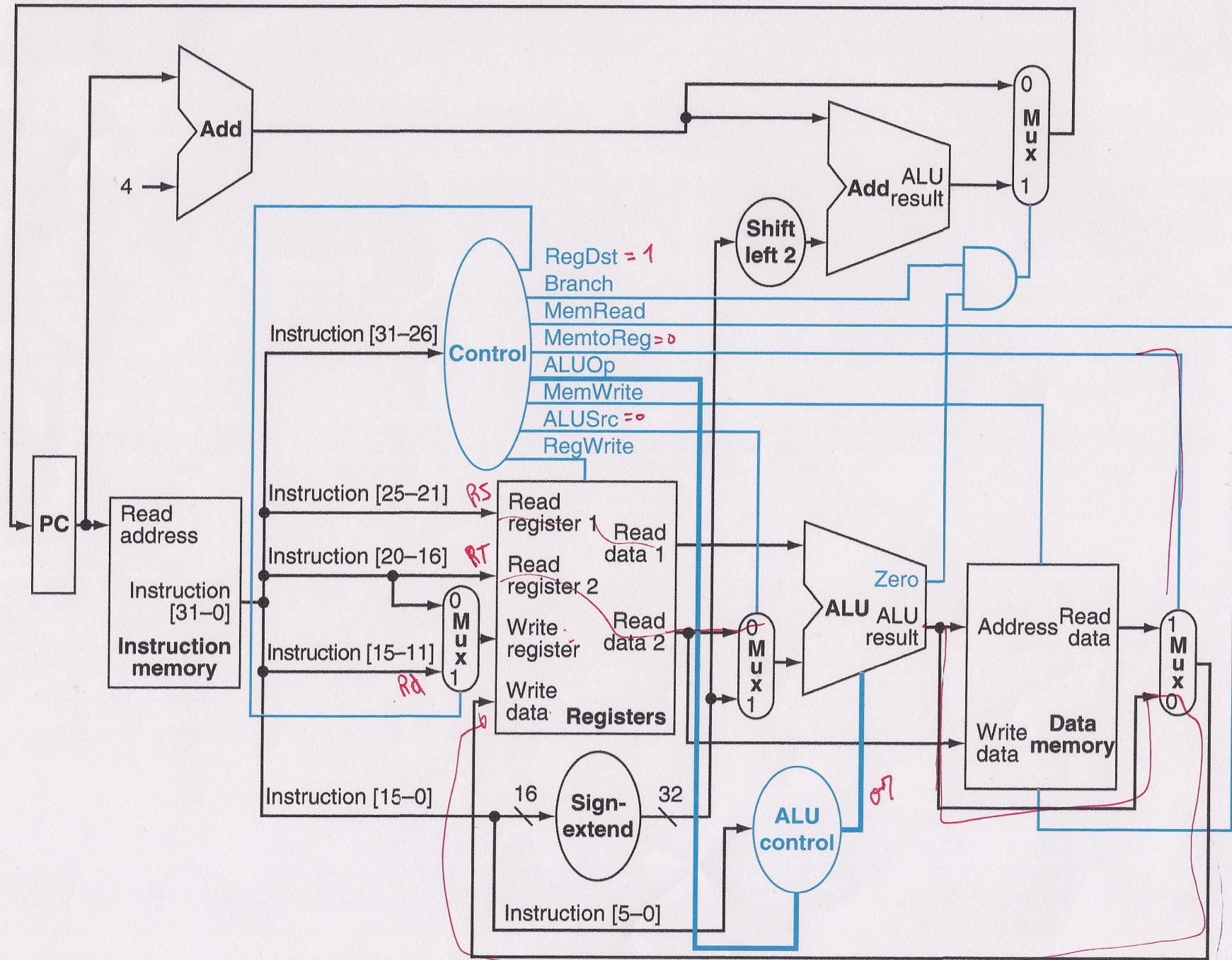




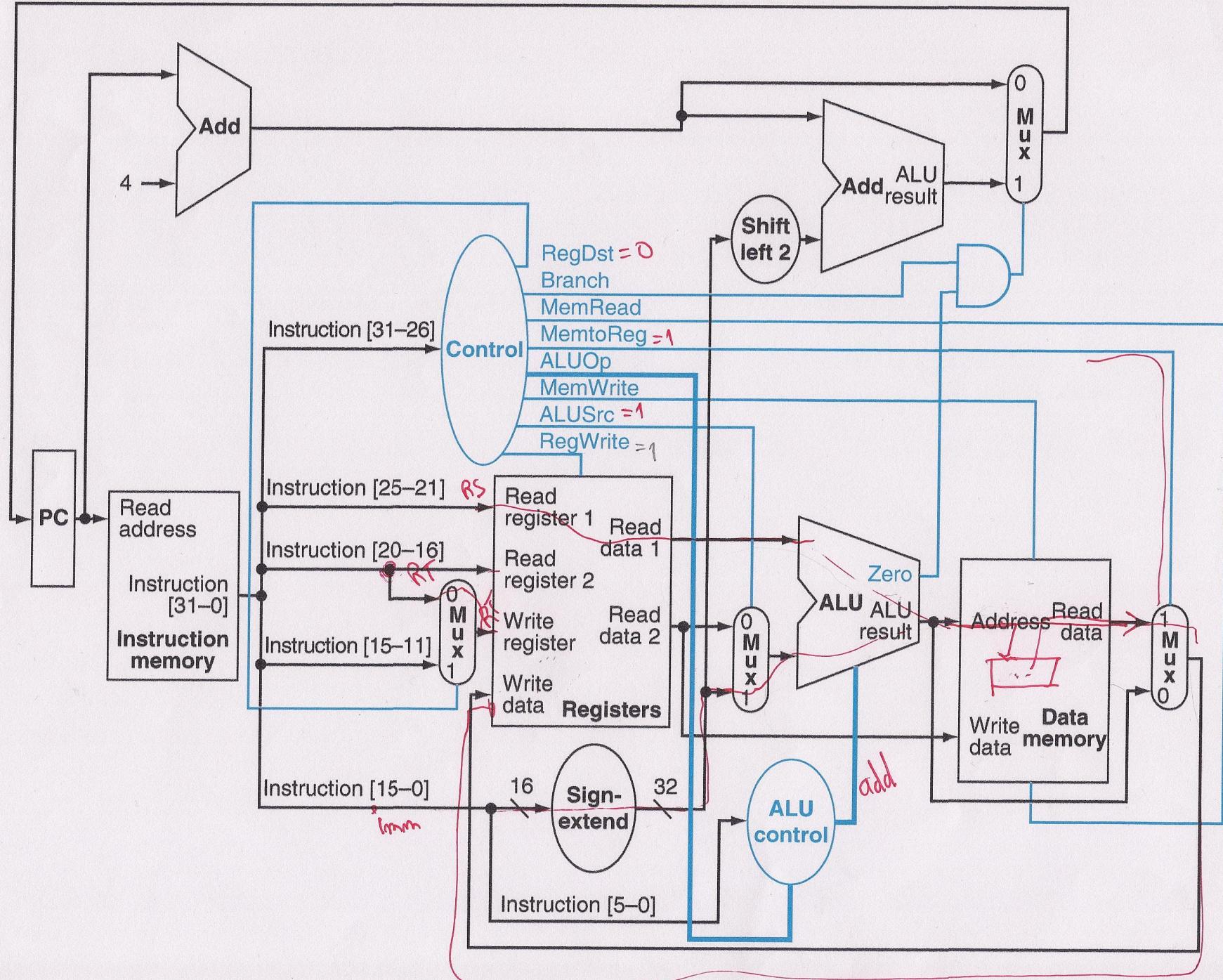
and Rd Rs Rt



$$01 = R_d \quad R_s \quad R_t$$

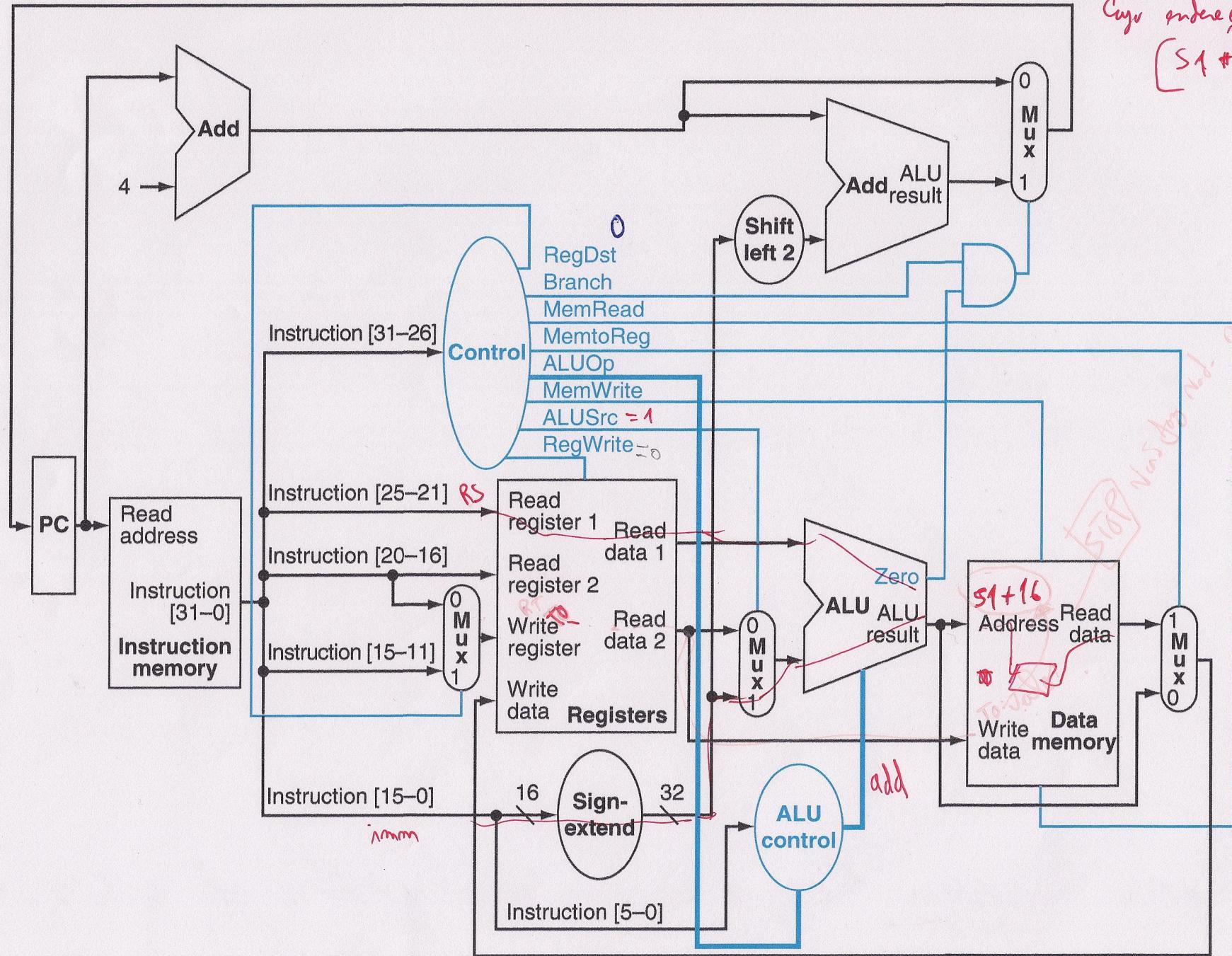


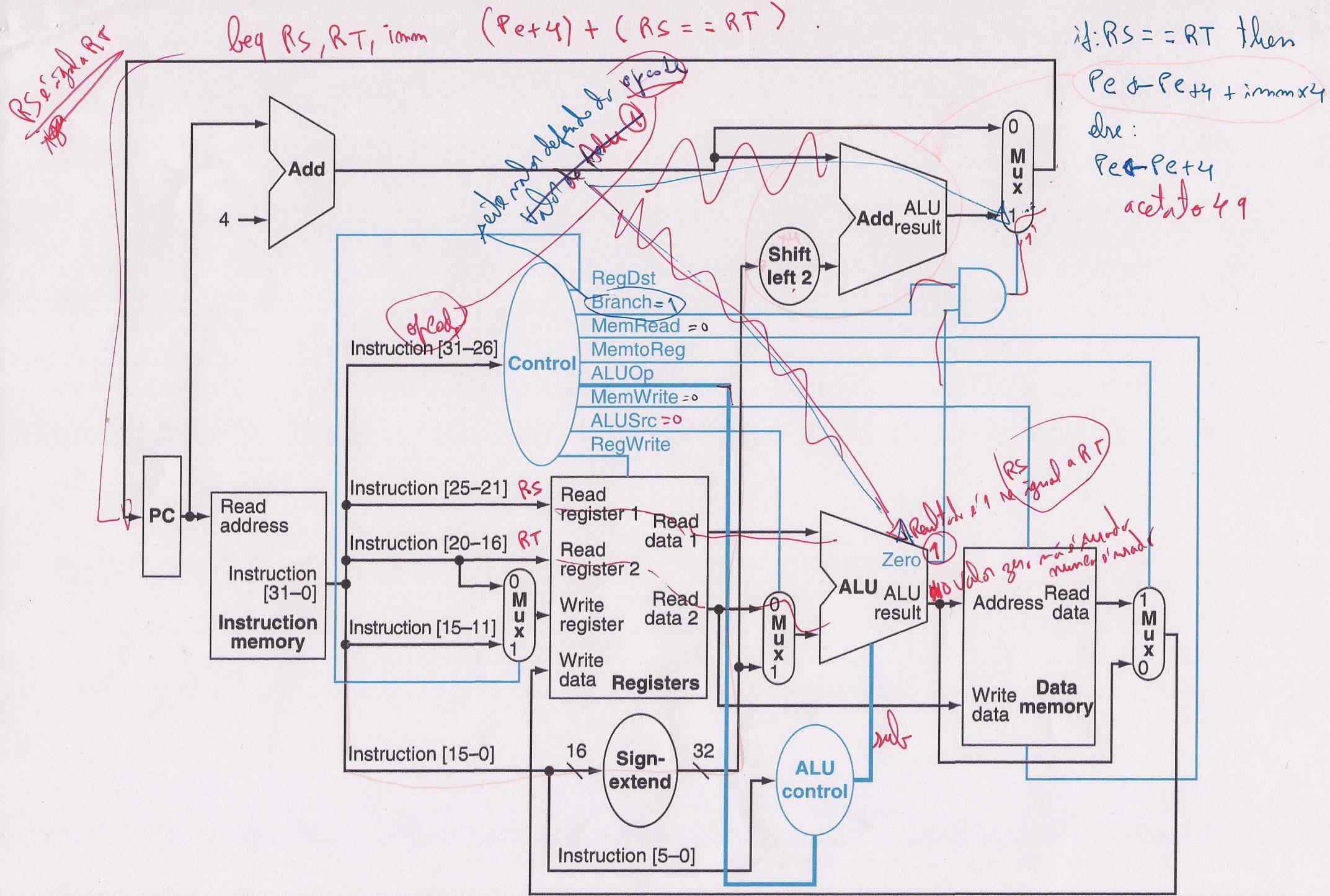
lw RT, imm(RS)



SW RT imm(RS)

SW To, 16 (S1)
Mem [S1 + 16] → To
guard against To no longer in memory
Copy address e'
[S1 + 16]





RS difusão de RT

beg RS RT imm

